CHALMERS
UNIVERSITY OF TECHNOLOGY

A greedy algorithm for optimal heating in powder-bed-based additive manufacturing

Journal of Mathematics in Industry
a SpringerOpen Journal

RESEARCH Open Access

A greedy algorithm for optimal heating in powder-bed-based additive manufacturing

Robert Forslund[1,2*], Anders Snis[1] and Stig Larsson[2]

*Correspondence: stig@chalmers.se
[1]Arcam EBM, Designvägen 2, SE–435 33 Mölnlycke, Sweden
[2]Mathematical Sciences, Chalmers University of Technology and University of Gothenburg, SE–412 96 Gothenburg, Sweden

Abstract

Powder-bed-based additive manufacturing involves melting of a powder bed using a moving laser or electron beam as a heat source. In this paper, we formulate an optimization scheme that aims to control this type of melting. The goal consists of tracking maximum temperatures on lines that run along the beam path. Time-dependent beam parameters (more specifically, beam power, spot size, and speed) act as control functions. The scheme is greedy in the sense that it exploits local properties of the melt pool in order to divide a large optimization problem into several small ones. As illustrated by numerical examples, the scheme can resolve heat conduction issues such as concentrated heat accumulation at turning points and non-uniform melt depths.

Keywords: Additive manufacturing; Powder bed fusion; Process control; Greedy optimization

1 Introduction

Powder bed fusion (PBF) is a type of additive manufacturing (AM) where metal powder is melted by a laser or electron beam in a layer-wise fashion to enable the production of geometrically complex parts [1]. AM undergoes continuous progress towards a technology that is robust and efficient, but there are still issues when it comes to quality and repeatability.

It is important for completed parts to meet the mechanical requirements and quality standards specified by the applications for which they are manufactured. The qualities of a completed part, such as tensile strength and surface roughness, depend on the melting process, which in turn is governed by several dozen material and process parameters such as preheating temperature, powder packing ratio, and beam speed, among other. Correlations between process parameters, process signatures (such as melt pool size and temperature), and product qualities are presented in [2] and references therein. Due to these correlations, the design of process settings requires critical attention. However, this involves extensive and costly experimental work and the resulting melting schemes implemented in machines rely on an excessive amount of parameters and functions in order to account for the dynamics of the melting process. This approach makes it difficult to optimize PBF and limits both the number of applications and the number of materials available for manufacturing.

As of late, computation based techniques are used to improve PBF related process control, i.e., to tackle the question of how to select different process parameters in order to build parts with desired properties [2, 3]. A common approach is design of (computational) experiments (DoE) [4, 5], which is an exploratory tool used to identify parameters that influence the qualities of the completed part. In [6], a DoE with a finite element model determines that the beam power and beam speed are the two process controllable parameters that have the largest impact on peak temperature during a single track melt. In [7], a DoE based on a simple thermal model [8] is used to determine optimal process parameters for building high density parts. Also this study shows that the beam power and beam speed have the largest impact on melt pool width and depth. Empirical modelling techniques such as artificial neural networks are also used to determine the optimal selection of process parameters, see, e.g., [9] as well as [10] and references therein.

While DoE is useful for developing a deeper understanding of the melting process, it can be a tedious affair to use them for optimization. An optimal control approach might be better suited for that purpose, as it starts off with preset, desired melt characteristics and seeks corresponding optimal process parameters via a mathematical optimization problem.

Attempts of optimizing melt pool characteristics have been made based on the two-phase Stefan problem, where the free boundary between solid and liquid is understood via the Stefan condition. In the context of PBF, the free boundary characterizes the size and shape of the melt pool. In [11], the two-phase Stefan problem in a container is considered, and the temperature on the container boundary is optimized with respect to a desired transient evolution of the free boundary. Optimization problems based on quasi-steady state formulations of the two-phase Stefan problem are solved in [12, 13]. Here the desired free boundary between liquid and melt is prescribed and the goal involves tracking the melt temperature on the prescribed free boundary.

This paper is part of an effort that aims to reduce the number of parameters needed in the design of melt schemes. We present a simulation based framework that seeks to facilitate process optimization and material development, while keeping computational costs at a minimum. Thus, computational efficiency is prioritized as we trade a certain level of detail for a fast optimization scheme that can be applied to the melting of large domains. The scheme is efficient for three reasons:

- The continuum thermal model for describing heat conduction includes an analytic solution that allows for fast and pointwise computation of temperatures during melting [14, 15]. The model assumes that the beam parameters are piecewise constant in time. We remark that beam parameters in actual AM machines are often set to vary in such a discontinuous way. The model does not include phase change, and does in particular not capture the solid-liquid interface. Instead, we use the term melt pool to simply denote the region where the temperature is larger than the melt temperature.
- The formulation of the optimization problem involves a severe model order reduction. Rather than striving for some desired temperature distribution that is difficult to express, the goal consists of tracking preset reference maximum temperatures on lines that run along the beam path.
- The resulting optimization problem is solved by a greedy algorithm that divides it into several small problems that are easier to solve. These sub-problems are solved consecutively as the beam traverses the powder bed.

Together, this framework comprises an optimization scheme for process control as suggested in [16].

More general approaches were tested and will be further explored in future work. For instance, a full finite element simulation of the heat equation combined with an adjoint-based optimization was implemented. However, this must be made more computationally efficient by appropriate simplifications before it can be used for process optimization. The proposed Greedy algorithm was chosen as a first attempt in this direction.

As noted earlier, DoE suggests that the beam parameters have the largest impact on the temperature distribution during melting and, ultimately, on the quality of the completed part. For this reason, we choose the beam power, spot size and speed as control variables. Our scheme allows for time-dependent beam parameters, which increases their ability to control the melting process. An ability to optimize these beam parameters is useful not only for validation, but also in order to speed up the development of process settings for new (and old) materials. Since optimization is an iterative process, the solver of the forward problem needs to be highly efficient. The analytic solution provides such efficiency.

The remainder of this paper is organized as follows. Section 2 describes the thermal model and its corresponding analytic solution. The optimization problem is formulated in Sect. 3. Section 4 and Sect. 5 propose two greedy algorithms for solving said problem. In Sect. 6 we apply our optimization scheme in numerical examples. Here we also detail how the combination of the two greedy algorithms can aid in the development of so called beam parameter functions. Additional comments and concluding remarks are given in Sect. 7.

2 Thermal model

Consider the heat equation on the lower half space $\Omega = \mathbb{R}^2 \times \mathbb{R}_-$ during a time span $\mathcal{T} = (0, T]$. Let Γ denote the surface boundary $z = 0$ and let u_{init} denote the constant initial temperature. The beam travels on the surface Γ along a preset, piecewise linear path

$$\mathcal{C}^{\text{s}} = \left\{ \boldsymbol{x}^{\text{s}}(t) : t \in \mathcal{T} \right\},$$

where $\boldsymbol{x}^{\text{s}}(t) = (x^{\text{s}}(t), y^{\text{s}}(t), 0)$ is the position of the center of the beam at time t. The heat flux Φ due to a scanning electron beam is modeled as a Gaussian function

$$\Phi = \Phi(x, y, t) = \frac{P(t)}{2\pi\sigma(t)^2} \exp\left(-\frac{(x - x^{\text{s}}(t))^2 + (y - y^{\text{s}}(t))^2}{2\sigma(t)^2} \right).$$

The three beam parameters are the absorbed beam power $P(t)$, the beam spot size $\sigma(t)$, and the beam speed $v(t) = |\boldsymbol{v}(t)|$. Here $\boldsymbol{v}(t) = (v_x(t), v_y(t), 0) = (v(t)\cos\theta(t), v(t)\sin\theta(t), 0)$, where $\theta(t)$ is the angle between the positive x-axis and the direction of the path. This angle is known for any t since the beam path $\mathcal{C}^{\text{s}}$ is preset and it follows that v uniquely defines the vector $(v_x, v_y, 0)$. The position of the beam $\boldsymbol{x}^{\text{s}}(t)$ depends on the speed with which the beam has traveled the path $\mathcal{C}^{\text{s}}$ up to time t. The beam parameters are often set to vary in a piecewise constant fashion in AM machines. The following definition makes the concept of piecewise constant beam parameters more precise.

Definition 1 Given times $0 = t_0 < t_1 < \ldots < t_N = T$,

$$(t_{n-1}, t_n] = \left(t_n^{\text{i}}, t_n^{\text{f}}\right], \quad n = 1, 2, \ldots, N,$$

is a partition of $\mathcal{T}$ consisting of N segments. Index n indicates the nth segment in the partition, and a segment in turn is a collection of the following data:

- t_n^{i}, t_n^{f}; an initial time and final time, respectively,
- (P_n, σ_n, v_n); a triplet of power, spot size, and speed such that

$$\big(P(t), \sigma(t), v(t)\big) = (P_n, \sigma_n, v_n) \quad \text{if } t \in \big(t_n^{\mathrm{i}}, t_n^{\mathrm{f}}\big],$$

- ℓ_n^{s}; a line traversed by the beam between times t_n^{i} and t_n^{f},
- $\boldsymbol{x}_n^{\mathrm{i}}$, $\boldsymbol{x}_n^{\mathrm{f}}$; the initial position and final position of ℓ_n^{s}, respectively.
- $\theta_n = \tan^{-1}\big(\frac{y_n^{\mathrm{f}} - y_n^{\mathrm{i}}}{x_n^{\mathrm{f}} - x_n^{\mathrm{i}}}\big)$; the angle between the positive x-axis and the direction of the path.

With $\hat{z}$ the outward unit normal of Ω, the heat transfer problem can be written as

$$\begin{aligned} &\rho c_p \frac{\partial u}{\partial t} - \nabla \cdot (\lambda \nabla u) = 0 \quad \text{in } \Omega \times \mathcal{T}, \\ &(\lambda \nabla u) \cdot \hat{z} = \Phi \quad \text{on } \Gamma \times \mathcal{T}, \\ &u(\cdot, 0) = u_{\mathrm{init}} \quad \text{in } \Omega, \end{aligned} \tag{1}$$

where ρ, c_p, λ denote density, heat capacity, and thermal conductivity, respectively. These material parameters are assumed to be constant. This gives us the thermal diffusivity $\kappa = \lambda/\rho c_p$. Problem (1) has an analytic solution [14, 15]. We refer to these sources for a detailed derivation of this solution and merely outline it here.

Proposition 1 *Given a partition as in Definition 1, the solution of problem (1) can be written as*

$$u(\boldsymbol{x}, t) = u_{\mathrm{init}} + \sum_{k=1}^{n-1} u_{n,k}^{\mathrm{I}}(\boldsymbol{x}, t) + u_n^{\Phi}(\boldsymbol{x}, t) \quad \textit{for } t \in \big(t_n^{\mathrm{i}}, t_n^{\mathrm{f}}\big], n = 1, 2, \ldots, N,$$

where $u_{n,k}^{\mathrm{I}}$ is the temperature due to the earlier scanning of segment $k < n$ and u_n^{Φ} is the temperature due to the current scanning of segment n.

The analytic expressions of $u_{n,k}^{\mathrm{I}}$ and u_n^{Φ} are derived in [15], wherein it also described how the solution can be efficiently computed.

The power is largely determined by the beam current, and this current can not be adjusted at a fast rate. Therefore, P is set to be constant for all $t \in \mathcal{T}$ for the remainder of this paper. It should be noted, however, that if one would be interested in optimizing all three beam parameters, the following extends to the case of non-constant power as well.

A remark on our thermal model is in order, as it does not include cooling, the latent heat of fusion nor a description of the solid-liquid intersection between powder and melt pool, which makes the notion of a melt pool quite fuzzy. Here we use the term melt pool to simply denote the volume where the temperature is larger than the melt temperature. Hence we use the isothermal $\{\boldsymbol{x}(t) : u(\boldsymbol{x}, t) = u_{\mathrm{melt}}\}$, where u_{melt} is the melt temperature of the powder, to represent the solid-liquid interface. Furthermore, since the model is a continuum model, it breaks down on the mesoscale where we see phenomena such as balling, inter-capillary effects, Plateau–Rayleigh instabilities, thermal expansion, among

other [17, 18]. Despite these restrictions, it is anticipated that effective parameters, tuned via comparisons with experiments, can be used to make the thermal model reliable enough for control and optimization. The optimization problem described below also aligns with the overarching aim to reduce the number of parameters needed for process control.

3 Formulation of the optimization problem

The goal is to optimize the melting process with respect to the beam spot size and beam speed. Since these beam parameters are defined in a piecewise constant fashion according to Definition 1, we can write

$$\sigma(t) = \sum_{k=1}^{N} \sigma_k \chi_{(t_k^{\mathrm{i}}, t_k^{\mathrm{f}}]},$$
$$\nu(t) = \sum_{k=1}^{N} \nu_k \chi_{(t_k^{\mathrm{i}}, t_k^{\mathrm{f}}]},$$

where χ is the indicator function. Since we aim to optimize the speed, either the times $t_n^{\mathrm{i}}, t_n^{\mathrm{f}}$ or the positions $\boldsymbol{x}_n^{\mathrm{i}}, \boldsymbol{x}_n^{\mathrm{f}}$, $n = 1, 2, \ldots, N$, in Definition 1 will have to vary during optimization. Due to the beam path being preset and reasons that will become clear in the next section, it is better to fix the positions. Therefore it is more appropriate to express σ and ν as space dependent functions instead. To this end, introduce the scanning distance

$$\gamma(\boldsymbol{x}^{\mathrm{s}}) = \sum_{k=1}^{n-1} \left|\boldsymbol{x}_k^{\mathrm{f}} - \boldsymbol{x}_k^{\mathrm{i}}\right| + \left|\boldsymbol{x}^{\mathrm{s}} - \boldsymbol{x}_n^{\mathrm{i}}\right|, \quad \text{if } \boldsymbol{x}^{\mathrm{s}} \in \ell_n^s, n = 1, 2, \ldots, N.$$

Then we have the following beam parameter functions:

$$\begin{aligned} \sigma\left(\gamma\left(\boldsymbol{x}^{\mathrm{s}}\right)\right) &= \sum_{k=1}^{N} \sigma_k \chi_{(\gamma(\boldsymbol{x}_k^{\mathrm{i}}), \gamma(\boldsymbol{x}_k^{\mathrm{f}})]}, \\ \nu\left(\gamma\left(\boldsymbol{x}^{\mathrm{s}}\right)\right) &= \sum_{k=1}^{N} \nu_k \chi_{(\gamma(\boldsymbol{x}_k^{\mathrm{i}}), \gamma(\boldsymbol{x}_k^{\mathrm{f}})]}. \end{aligned} \tag{2}$$

From (2), a decision vector can immediately be extracted as $\boldsymbol{d} = (\boldsymbol{\sigma}, \boldsymbol{\nu}) = \{(\sigma_k, \nu_k)\}_{k=1}^{N} = (\sigma_1, \nu_1, \sigma_2, \nu_2, \ldots, \sigma_N, \nu_N)$. The variables in the decision vector are bounded due to practical limitations, $\boldsymbol{d}_{\min} \leq \boldsymbol{d} \leq \boldsymbol{d}_{\max}$.

The control of the melting process is a multiobjective optimization problem due to the many correlations between process parameters, process signatures, and product qualities. The qualities of the final part are strongly dependent on the temperatures obtained during the melting process [19]. The characteristics of the melt pool are important process signatures. If the melt pool is too small relative to the line offset (i.e., the distance between two hatch lines) and layer depth, powder might be left unmelted between hatch lines or between layers, causing discontinuities and porosity. Furthermore, high surface temperatures might result in too much evaporation and subsequent recoil pressure, which can result in undesired material transport such as ejection of molten materials that later cause defects [20] or formations of small ridges that prohibit the deposition of new powder layers and thus cause the manufacturing process to terminate [21]. Qualitatively, therefore, the choice of cost functional can be motivated by the desire to

1. maintain a uniform and appropriately sized melt pool during melting, and
2. avoid too high surface temperatures.

3.1 A reductive approach

Consider a beam scanning along a path $\mathcal{C}^{\mathrm{s}}$. Denote by ω a supposed desired melt pool

$$\omega(t) = \left\{\boldsymbol{x} \in \Omega : u(\boldsymbol{x}, t) \geq u_{\mathrm{melt}}\right\}.$$

It is difficult to express $\omega(t)$ explicitly. Instead, we consider the final solidified volume. With our purely thermal model, the powder-solid interface of this volume is dependent on the maximum temperature and would be easier to explicitly define than the melt pool $\omega(t)$. However, even further reductions can be made by isolating particular curves on this powder-solid interface. More precisely, we introduce a secondary path $\mathcal{C}^{\mathrm{wd}}$ chosen such that it lies on a desired powder-solid interface. The secondary path is related to the beam path by some function $\mathfrak{F} : \mathcal{C}^{\mathrm{s}} \to \mathcal{C}^{\mathrm{wd}}$ and we write

$$\mathcal{C}^{\mathrm{wd}} = \left\{\mathfrak{F}\left(\boldsymbol{x}^{\mathrm{s}}(t)\right) : t \in \mathcal{T}\right\}$$

and let $\boldsymbol{x}^{\mathrm{wd}} = \mathfrak{F}(\boldsymbol{x}^{\mathrm{s}})$. For example, if the beam path consists of one segment, then a simple example of a secondary path is

$$\mathcal{C}^{\mathrm{wd}} = \left\{\left(x^{\mathrm{s}}(t) + w\sin\theta_1,\, y^{\mathrm{s}}(t) - w\cos\theta_1,\, -d\right) : t \in \mathcal{T}\right\}.$$

The idea is that the secondary path relates to $\mathcal{C}^{\mathrm{s}}$ via a width w and a depth d. With this, the description of the optimal melting reduces to two paths; $\mathcal{C}^{\mathrm{s}}$ (preset) and $\mathcal{C}^{\mathrm{wd}}$ (chosen with respect to $\mathcal{C}^{\mathrm{s}}$). This reductive approach is illustrated in Fig. 1.

As we shall see in the following section, the steps taken above allow us to formulate a simple optimization problem that is efficient in the sense that we, instead of tracking some desired transient melt pool $\omega(t)$ in a volume, only track two scalar values u_{melt} and u_{surf} on paths.

3.2 Mathematical formulation

Following the reduction in Sect. 3.1, we are now interested in maximum temperatures on paths running along the beam path since these temperatures determine the size of the subsequent solidified volume. Before we formulate the problem, we need the following.

Definition 2 (Hatch line) Given a partition as in Definition 1, two segments k and $k+1$, $1 \leq k < N-1$, are connected if $\boldsymbol{x}_k^{\mathrm{f}} = \boldsymbol{x}_{k+1}^{\mathrm{i}}$. A sequence $\{i\}_{i=k}^{K}$ of connected segments form a hatch line if $\theta_n = \theta_m\ \forall n, m \in [k, K]$ and $\theta_{k-1} \neq \theta_k$ and $\theta_K \neq \theta_{K+1}$.

The total number of hatch lines M satisfies $1 \leq M \leq N$.

Define also the maximum temperature field

$$\mathcal{M}(\boldsymbol{x}; \boldsymbol{d}) = \max_{t \in \mathcal{T}}\left\{u(\boldsymbol{x}, t; \boldsymbol{d})\right\}.$$

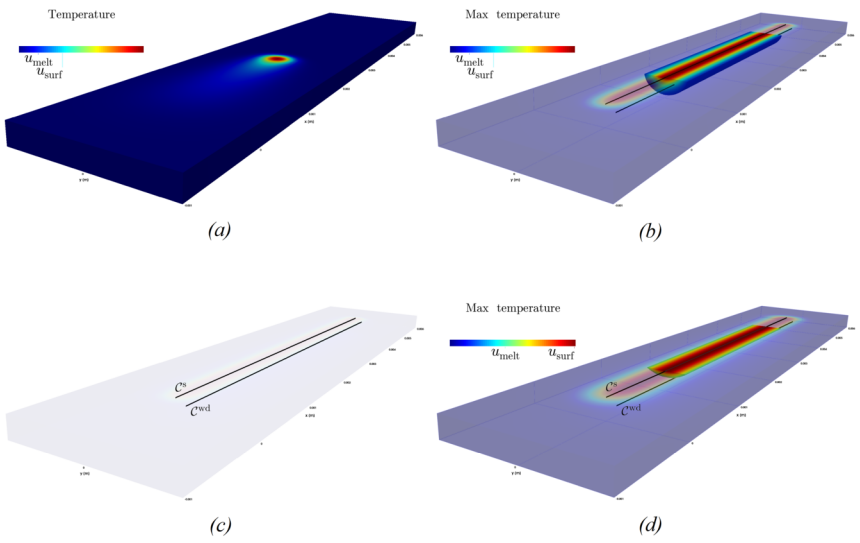


Figure 1 The desire to optimize the size and shape of the melt pool is reduced to a problem of tracking maximum temperatures on paths. (a) Temperature distribution due to a moving beam. (b) Maximum temperature obtained during melting, with the solidified volume highlighted. (c) Introduction of beam path $\mathcal{C}^{\text{s}}$ and secondary path $\mathcal{C}^{\text{wd}}$. The secondary path is drawn along the desired liquid-solid interface. (d) Maximum temperature after optimization, with the solidified volume highlighted. The shape of this volume is optimized due to the tracking of reference temperatures u_{surf} and u_{melt} on $\mathcal{C}^{\text{s}}$ and $\mathcal{C}^{\text{wd}}$, respectively.

We want $\mathcal{M}(\boldsymbol{x};\boldsymbol{d}) = u_{\text{melt}}$ for all $\boldsymbol{x}$ on $\mathcal{C}_i^{\text{wd}}$ in order to ensure a uniform and thorough melting. Similarly, $\mathcal{M}(\boldsymbol{x};\boldsymbol{d})$ should not exceed some maximum allowed temperature u_{surf} on $\mathcal{C}^{\text{s}}$.

The resulting objective vector becomes

$$f(\boldsymbol{d}) = \{f_1(\boldsymbol{d}), f_2(\boldsymbol{d})\}, \tag{3}$$

where

$$\begin{aligned} f_1(\boldsymbol{d}) &= \int_{\mathcal{C}^{\text{wd}}} \alpha(\boldsymbol{x}^{\text{wd}}) \cdot \left(\mathcal{M}(\boldsymbol{x}^{\text{wd}};\boldsymbol{d}) - u_{\text{melt}}\right)^2 \mathrm{d}x, \\ f_2(\boldsymbol{d}) &= \int_{\mathcal{C}^{\text{s}}} \alpha(\boldsymbol{x}^{\text{s}}) \cdot \left(\mathcal{M}(\boldsymbol{x}^{\text{s}};\boldsymbol{d}) - u_{\text{surf}}\right)^2 \mathrm{d}x. \end{aligned} \tag{4}$$

Here

$$\alpha(\boldsymbol{x}) = \begin{cases} 0 & \text{if } \boldsymbol{x} \text{ is near the start or end of a hatch line,} \\ 1 & \text{otherwise} \end{cases} \tag{5}$$

is a weight that excludes intervals from the cost functional if they are very close to the start point or end point of a hatch line. This type of weight is inserted because the total heat supplied to a region near a start point, for instance, is comparatively small since the beam only moves away from it rather than passing it. As a consequence, it can be difficult to reach the reference temperatures in these regions and if included, they deteriorate the overall performance of the optimizer. Therefore, it is better to ignore these intervals in

the goal functional and instead let them be covered by the contouring stage, in which the beam scans along the boundary of the shape being melted. The contouring stage also improves the surface finish of the part [22]. Effectively, this choice of α simply means that the domains of integration in (4) become slightly smaller.

The functionals (4) are of tracking type where we use the L_2 norm to minimize the distances between the actual maximum temperature and the desired maximum temperatures. The reason for tracking these temperatures rather than putting constraints on the temperature is that we want to control the shape and size of the melt pool not the temperature itself. The reason for tracking also the surface temperature is that it helps restricting the shape of the melt pool. Without this restriction, one could potentially end up with an extremely wide and shallow type of melting, for instance. However, it leads to multiobjective optimization. We also note that the functionals (4) are nondifferentiable, because $\mathcal{M}(\boldsymbol{x};\boldsymbol{d})$ is not differentiable with respect to u due to the evaluation of the maximum. Both these facts make the resulting optimization problem more difficult. We show in Sect. 6 how these difficulties can be overcome.

The analytic solution presented in Sect. 2 allows for pointwise computations of temperatures, and that is why we can easily compute temperatures on lines, which saves a large amount of computation time compared to doing so on surfaces or in volumes. This remark highlights a big motivation behind the reduction carried out in Sect. 3.1.

The objective vector (3) needs to be translated into a scalar valued cost functional in order to use standard nonlinear programming solvers. We use a scalarization known as the weighting method. In this method the weighted sum of the objectives is minimized. We introduce weights $W_i \geq 0$, $i = 1, 2$. The scalarized optimization problem becomes

$$\begin{aligned} &\text{minimize} \quad J(\boldsymbol{d}) = W_1 f_1(\boldsymbol{d}) + W_2 f_2(\boldsymbol{d}) \\ &\quad \text{subject to} \\ &\qquad \text{state eq. (1)}, \qquad \text{(PDE constraint)} \\ &\qquad \boldsymbol{d}_{\min} \leq \boldsymbol{d} \leq \boldsymbol{d}_{\max}. \qquad \text{(Parameter constraint)} \end{aligned} \tag{6}$$

The choice of weights should represent the relative importance of the objectives; important objectives are weighted more heavily.

4 A first greedy algorithm for solving the scalarized optimization problem

In order to speed up the optimization, we propose a method that makes use of the fact that the melt pool, and hence maximum temperatures, are localized to the beam. The proceeding involves a division of $\mathcal{T}$ into subintervals on the form $\bigcup_{i=p}^{q}(t_i^{\mathrm{i}}, t_i^{\mathrm{f}}]$. Local optimization problems are solved on these subintervals and optimal parameter pairs (σ_p, ν_p) are frozen sequentially. When given parameter pair(s) has been frozen, the local problem is translated in time (and space) and the initial condition is updated. As such, this greedy type of algorithm divides the optimization problem (6) into several smaller optimization problems that are faster to solve.

The goal functional in (6) involves maximum temperatures over time near the beam path, which is a property that is local to the beam itself. Given a point P on, say, $\mathcal{C}^{\mathrm{s}}$, it is known that P will obtain its largest temperature during a time window when the beam, and the melt pool it generates, passes P. Therefore, it is the values of the beam parameters

during this particular time window that has the highest influence on maximum temperature at P. The reasoning is similar for a point on $\mathcal{C}^{\mathrm{wd}}$, the only difference being that it takes slightly longer to reach the maximum temperature on $\mathcal{C}^{\mathrm{wd}}$ since heat diffusion is not an instantaneous process. Therefore, we decide to split the optimization problem (6) into multiple subproblems that are solved sequentially in time while parameter pairs are frozen as we go along.

In order to formalize the method, we make the following definition.

Definition 3 (Window). Given a partition as in Definition 1, a time window $\mathcal{T}_{p,q} = \cup_{i=p}^{q}(t_i^{\mathrm{i}}, t_i^{\mathrm{f}}]$ is defined as a set of adjacent segments in $\mathcal{T}$. The size of $\mathcal{T}_{p,q}$ is the number of segments that constitutes it. The local beam path and local secondary path corresponding to $\mathcal{T}_{p,q}$ are given by

$$\mathcal{C}_{p,q}^{\mathrm{s}} = \left\{\boldsymbol{x}^{\mathrm{s}}(t) : t \in \mathcal{T}_{p,q}\right\},$$
$$\mathcal{C}_{p,q}^{\mathrm{wd}} = \left\{\boldsymbol{x}^{\mathrm{wd}}(t) : t \in \mathcal{T}_{p,q}\right\}.$$

Hence the beam traverses the path $\mathcal{C}_{p,q}^{\mathrm{s}}$ during time $\mathcal{T}_{p,q}$.

Define a local decision vector $\boldsymbol{d}_{p,q} = \{(\sigma_k, v_k)\}_{k=p}^{q}$ and a local maximum temperature field

$$\mathcal{M}_{p,q}(\boldsymbol{x}; \boldsymbol{d}_{p,q}) = \max_{t \in \mathcal{T}_{p,q}} \left\{u(\boldsymbol{x}, t; \boldsymbol{d}_{p,q})\right\}.$$

Similarly, we define the local objectives

$$g_1(\boldsymbol{d}_{p,q}) = \int_{\mathcal{C}_{p,q}^{\mathrm{wd}}} \beta_{p,q}\left(\boldsymbol{x}^{\mathrm{wd}}\right) \cdot \alpha\left(\boldsymbol{x}^{\mathrm{wd}}\right) \cdot \left(\mathcal{M}_{p,q}\left(\boldsymbol{x}^{\mathrm{wd}}; \boldsymbol{d}_{p,q}\right) - u_{\mathrm{melt}}\right)^2 \mathrm{d}s,$$
$$g_2(\boldsymbol{d}_{p,q}) = \int_{\mathcal{C}_{p,q}^{\mathrm{s}}} \beta_{p,q}\left(\boldsymbol{x}^{\mathrm{s}}\right) \cdot \alpha\left(\boldsymbol{x}^{\mathrm{s}}\right) \cdot \left(\mathcal{M}_{p,q}\left(\boldsymbol{x}^{\mathrm{s}}; \boldsymbol{d}_{p,q}\right) - u_{\mathrm{surf}}\right)^2 \mathrm{d}s.$$

Here $\beta_{p,q}$ is used to prioritize minimization of the errors over the earlier segments in the window. This weight accentuates the error on the segment(s) that is about to become frozen and it plays a crucial role; since the greedy algorithm never returns to a segment once it has been frozen it is important that the solver prioritizes this segment. Here we let $\beta_{p,q}$ be piecewise constant over the segments and determined by a function that decreases quadratically along $\mathcal{C}_{p,q}^{\mathrm{s}}$. See Fig. 2. Formally, we have

$$\beta_{p,q}\left(\boldsymbol{x}^{\mathrm{s}}\right) = \chi_{(\gamma(\boldsymbol{x}_p^{\mathrm{i}}), \gamma(\boldsymbol{x}_{p+r}^{\mathrm{i}})]} + \sum_{k=p+r}^{q-1} \left(\frac{\gamma(\boldsymbol{x}_q^{\mathrm{f}}) - \gamma(\boldsymbol{x}_k^{\mathrm{i}})}{\gamma(\boldsymbol{x}_q^{\mathrm{f}}) - \gamma(\boldsymbol{x}_p^{\mathrm{k}})}\right)^2 \chi_{(\gamma(\boldsymbol{x}_k^{\mathrm{i}}), \gamma(\boldsymbol{x}_k^{\mathrm{f}})]}$$

on $\mathcal{C}_{p,q}^{\mathrm{s}}$. We define $\beta_{p,q}(\boldsymbol{x}^{\mathrm{wd}})$ in a similar fashion. The choice of a quadratic underlying function is based on tests that investigate how the weight affects the optimization results.

Input : A partition as in Definition 1.
Remaining optimization problem data ($\mathcal{C}^{\text{wd}}, u_{\text{melt}}, u_{\text{surf}}$, bounds, weights).

Parameters: $(\boldsymbol{\sigma}, \boldsymbol{\nu}) = \{(\sigma_k, \nu_k)\}_{k=1}^{N}$ ▷ Parameter pairs/decision array.
$p = 1$ ▷ Index of first segment in current window.
$q \in \{1, \dots, N\}$ ▷ Index of last segment in current window.
$r \in \{1, \dots, q - p + 1\}$ ▷ Number of segments to freeze next.

Output : $(\boldsymbol{\sigma}^{\text{opt}}, \boldsymbol{\nu}^{\text{opt}})$, J^{opt} ▷ Optimal decision vector and objective with respect to the subproblems (7). (We cannot expect to find the optimal decision vector for the global problem (6), only an approximation of it.)

begin
  while $p \leq N$ **do**
    Solve subproblem (7) for window $\mathcal{T}_{p,q}$ with initial guess $\boldsymbol{d}_{p,q} = \{(\sigma_k, \nu_k)\}_{k=p}^{q}$ to find candidate decision variables $\{(\tilde{\sigma}_k, \tilde{\nu}_k)\}_{k=p}^{q}$.
    $$(\sigma_k, \nu_k) \leftarrow \begin{cases} (\tilde{\sigma}_k, \tilde{\nu}_k), & k = p, \dots, q \\ (\tilde{\sigma}_q, \tilde{\nu}_q), & k = q+1, \dots, N \end{cases}$$ ▷ Update parameter pairs.
    $(\sigma_k^{\text{opt}}, \nu_k^{\text{opt}}) = (\sigma_k, \nu_k), \quad k = p, \dots, p + r - 1$ ▷ Freeze r parameter pairs.
    Change window location:
    $p \leftarrow p + r$ ▷ Update first index.
    $q \leftarrow \min\{q(p) + r, N\}$ ▷ Update last index.
    $r \leftarrow \min\{r(p), q - p + 1\}$ ▷ Update number of segments to freeze next.
  end
  $J^{\text{opt}} = J((\boldsymbol{\sigma}^{\text{opt}}, \boldsymbol{\nu}^{\text{opt}}))$ ▷ Compute optimal objective in (6).
end

Algorithm 1: Greedy algorithm for finding an approximate solution of the scalarized problem (6). Note that $q - p + 1$ equals the size of the current window.

Now, by employing the same scalarization as for the global problem (6), the resulting scalarized subproblem becomes

$$\begin{aligned} &\text{minimize} \quad J_{p,q}(\boldsymbol{d}_{p,q}) = W_1 g_1(\boldsymbol{d}_{p,q}) + W_2 g_2(\boldsymbol{d}_{p,q}) \\ &\quad \text{subject to} \\ &\qquad \text{state eq. (1)}, \qquad \text{(PDE constraint)} \\ &\qquad \boldsymbol{d}_{p,q_{\min}} \leq \boldsymbol{d}_{p,q} \leq \boldsymbol{d}_{p,q_{\max}}. \qquad \text{(Parameter constraint)} \end{aligned} \tag{7}$$

We can now formulate the greedy algorithm. This is done in Algorithm 1, and some complementary comments are given below. An illustration of the main idea is given in Fig. 3.

$\mathscr{L}4$: Future parameter pairs are updated as well, because the values found in the current window are likely a better guess than the initial one.

$\mathscr{L}5$: The segments corresponding to the frozen parameter pairs are removed from the window. In the implementation, certain checks can be made to determine

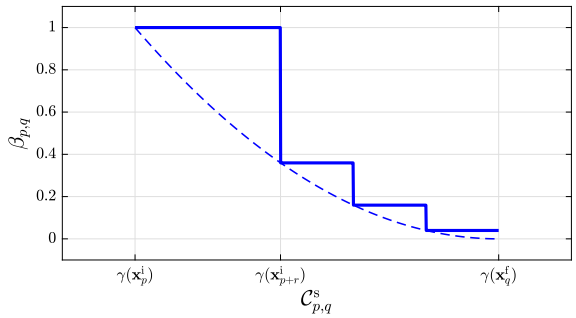


Figure 2 The weight $\beta_{p,q}$ is based on a quadratic function that decreases along the local beam path $\mathcal{C}_{p,q}^{\mathrm{s}}$. It is piecewise constant, largest on the first r segments in the window since they are about to be frozen (see Algorithm 1), and takes different values over the remaining segments. The weight is identical for the local secondary path $\mathcal{C}_{p,q}^{\mathrm{wd}}$.

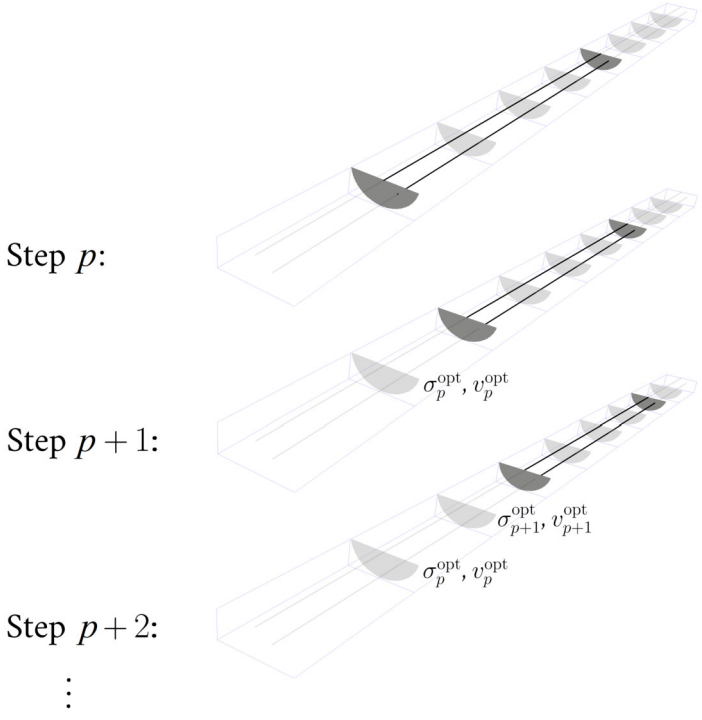


Figure 3 The greedy algorithm divides the optimization problem (6) into several smaller optimization problems that are easier to solve. The procedure involves a division of the beam path into subintervals. Localized optimization problems are solved on these intervals and optimal parameters are frozen in steps. After each such step, a new sub-problem is created by translating in time (ans space) and updating the initial condition. Here $q = p + 3$ and $r = 1$ (see Algorithm 1).

whether freezing should take place or not (as in, the amount of pairs to freeze). We leave out the details.

$\mathscr{L}$7-9: The window size is updated in preparation for the next iteration. The min operator is used to handle the ending when $q = N$. Note also that the parameters q and r may depend on p (i.e., on the location of the window). For instance, in a region where the beam path is complex or where the lengths of the segments are small,

we might require a large window size $q-p+1$, and hence a large q. Furthermore, the size of the melt pool should be taken into account when choosing the value of q.

The presented greedy algorithm can be utilized as a standalone tool for process optimization. If the beam path consists of N segments, the total number of parameters to optimize becomes $2N$. Now, depending on the design of the layer being melted, the value of N may be very high. In the next section we present a second version of the greedy algorithm that has the ability to significantly lower the size of the decision vector.

5 A second greedy algorithm based on fitting beam parameter functions

The greedy algorithm of this section expands on an example in [15]. As we shall see, it is similar to Algorithm 1 in most regards, but it is based on educated guesses of how the beam parameters should behave.

Recall from (2) the expression for piecewise constant beam parameters. The subsequent optimization makes no assumption on the behavior of the beam parameters along the beam path. An alternative approach is to do curve fitting of prespecified beam parameter functions. To this end, we write

$$\begin{aligned} \sigma\left(\boldsymbol{x}^{\mathrm{s}}\right) &= \sum_{k=1}^{N} F^{\sigma}\left(\boldsymbol{x}_k^{\mathrm{i}};\boldsymbol{\Lambda}^{\sigma}\right)\chi_{(\gamma(\boldsymbol{x}_k^{\mathrm{i}}),\gamma(\boldsymbol{x}_k^{\mathrm{f}})]}, \\ \nu\left(\boldsymbol{x}^{\mathrm{s}}\right) &= \sum_{k=1}^{N} F^{\nu}\left(\boldsymbol{x}_k^{\mathrm{i}};\boldsymbol{\Lambda}^{\nu}\right)\chi_{(\gamma(\boldsymbol{x}_k^{\mathrm{i}}),\gamma(\boldsymbol{x}_k^{\mathrm{f}})]}, \end{aligned} \tag{8}$$

where $\boldsymbol{\Lambda} = (\boldsymbol{\Lambda}^{\sigma}, \boldsymbol{\Lambda}^{\nu})$, the coefficients in our beam parameter functions, become our new decision vector that we want to optimize. Given a decision vector, the beam parameters are then evaluated as (see (2))

$$\begin{aligned} \sigma_k &= F^{\sigma}\left(\boldsymbol{x}_k^{\mathrm{i}};\boldsymbol{\Lambda}^{\sigma}\right), \\ \nu_k &= F^{\nu}\left(\boldsymbol{x}_k^{\mathrm{i}};\boldsymbol{\Lambda}^{\nu}\right). \end{aligned}$$

These parameter functions are defined with respect to the hatch lines as

$$\left.\begin{aligned} F^{\sigma}(\boldsymbol{x}^{\mathrm{s}};\boldsymbol{\Lambda}^{\sigma}) &= F_l^{\sigma}(\boldsymbol{x}^{\mathrm{s}};\boldsymbol{\Lambda}_l^{\sigma}) \\ F^{\nu}(\boldsymbol{x}^{\mathrm{s}};\boldsymbol{\Lambda}^{\nu}) &= F_l^{\nu}(\boldsymbol{x}^{\mathrm{s}};\boldsymbol{\Lambda}_l^{\nu}) \end{aligned}\right\} \quad \text{if } \boldsymbol{x}^{\mathrm{s}} \text{on hatch line } l.$$

The reason for splitting F^{σ} and F^{ν} between hatch lines is that a new hatch line often requires a rapid jump in beam parameter values.

It follows from Definition 2 that hatch lines can simply be seen as an intermediate level between the segments and the beam path. Let $\mathcal{S}:\{1,\ldots,M\}\to\{1,\ldots,N\}$ be an injective function that, given a hatch line m, returns the first segment in m. Let $\mathcal{L}:\{1,\ldots,N\}\to\{1,\ldots,M\}$ be a surjective function that, given segment n, returns the hatch line that contains it. With these two functions it is possible to seamlessly work with both hatch lines and segments. For instance, the local beam path that consists of hatch lines 1 to 3 is $\mathcal{C}^{\mathrm{s}}_{\mathcal{S}(1),\mathcal{S}(4)-1}$.

In terms of implementation, the second greedy algorithm is in many ways similar to the first greedy algorithm from the previous Sect. 4. They both rely on Definition 1 and piecewise constant beam parameters and they both solve subproblems of the form (7). What

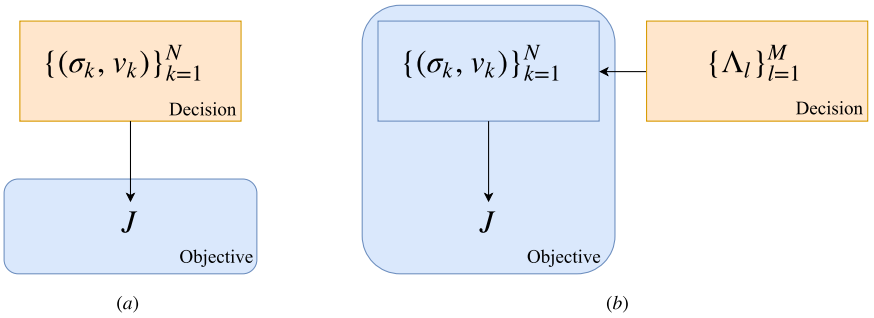


Figure 4 Difference between the two greedy algorithms. In the second version (b), the additional step significantly reduces the dimension of the decision vector if the number of lines is much smaller than the number of segments, i.e., if $M \ll N$.

separates them is the content of the decision vector $\boldsymbol{d}$ as illustrated in Fig. 4. In essence, the first algorithm optimizes the beam parameters *segmentwise* while the second algorithm optimizes the beam parameters *linewise*. The second greedy algorithm is detailed in Algorithm 2.

6 Numerical examples and discussion

We apply the greedy algorithm on a couple of single layer problems. The scalarized subproblems (7) are solved with the L-BFGS-B optimization algorithm [23, 24] provided by the optimization package of SciPy [25]. Given some iterate $\hat{\boldsymbol{d}}_{p,q}$, `scipy.optimize` approximates the gradient of $J_{p,q}(\hat{\boldsymbol{d}}_{p,1})$ using a 2-point finite difference estimation. Then L-BFGS-B, which is a quasi-Newton method, approximates the Hessian that enters in the local quadratic approximation of $J_{p,q}(\hat{\boldsymbol{d}}_{p,q})$. The solver options are selected to fit the scale of the problems considered here.

It is worth noting that the objective $J_{p,q}$ is not differentiable. At an initial stage, not only L-BFGS-B but also some gradient free methods were tested, and L-BFGS-B performed the best out of all solvers in those trials. It is not remarkable that L-BFGS-B performs well on nonsmooth problems as well (although we can not expect the same convergence as for a smooth optimization problem) [26]. On a related note, in the implementation we relax the scalarized subproblem (7) somewhat by replacing the (local) maximum temperature field $\mathcal{M}_{p,q}(\boldsymbol{x}; \boldsymbol{d}_{p,q})$ with an approximation,

$$\mathcal{M}_{p,q}(\boldsymbol{x}; \boldsymbol{d}_{p,q}) \approx \frac{1}{K} \log\left(\int_{\mathcal{T}_{p,q}} \exp\big(K \cdot u(\boldsymbol{x}, s; \boldsymbol{d}_{p,q})\big)\, \mathrm{d}s \right),$$

for an appropriate scalar K, which improves performance slightly. Finally, while the choice of starting point/initial decision vector can have a large impact on the performance of the optimizer, efforts related to this choice are not the main focus here and so disregarded.

It is important to emphasize that for practical use of the optimization scheme, the material data that enter the thermal model need to be fit with respect to experiments or a more detailed model. The determination of effective parameters is crucial since the model is simple and based on several assumptions. In the following examples we use material parameters that represent Ti-6Al-4V.

Input : A partition as in Definition 1; M hatch lines.
Remaining optimization problem data ($\mathcal{C}^{\text{wd}}$, u_{melt}, u_{surf}, bounds, weights).

Parameters: $\boldsymbol{\Lambda} = \{\boldsymbol{\Lambda}_l\}_{l=1}^{M}$ ▷ Coefficient sets/decision array.
$p = 1$ ▷ Index of first hatch line in current window.
$q \in \{1, \dots, M\}$ ▷ Index of last hatch line in current window.
$r \in \{1, \dots, q - p + 1\}$ ▷ Number of hatch lines to freeze next.

Output : $\boldsymbol{\Lambda}^{\text{opt}}$, J^{opt} ▷ Optimal decision vector and objective with respect to the subproblems (7). (We cannot expect to find the optimal decision vector for the global problem (6), only an approximation of it.)

begin
  while $p \leq N$ **do**
    Solve subproblem (7) for window $\mathcal{T}_{\mathcal{S}(p),\mathcal{S}(q+1)-1}$ with initial guess $\boldsymbol{d}_{\mathcal{S}(p),\mathcal{S}(q+1)-1} = \{\boldsymbol{\Lambda}_l\}_{l=p}^{q}$ to find candidate decision variables $\{\tilde{\boldsymbol{\Lambda}}_l\}_{l=p}^{q}$.
    $\boldsymbol{\Lambda}_l \leftarrow \begin{cases} \tilde{\boldsymbol{\Lambda}}_l, & l = p, \dots, q \\ \tilde{\boldsymbol{\Lambda}}_q, & l = q+1, \dots, M \end{cases}$ ▷ Update parameter pairs.
    $\boldsymbol{\Lambda}_l^{\text{opt}} = \boldsymbol{\Lambda}_l,\ l = p, \dots, p + r - 1$ ▷ Freeze r parameter pairs.
    Change window location:
    $p \leftarrow p + r$ ▷ Update first index.
    $q \leftarrow \min\{q(p) + r, N\}$ ▷ Update last index.
    $r \leftarrow \min\{r(p), q - p + 1\}$ ▷ Update number of segments to freeze next.
  end
  $J^{\text{opt}} = J(\boldsymbol{\Lambda}^{\text{opt}})$ ▷ Compute optimal objective in (6).
end

Algorithm 2: Second greedy algorithm for finding an approximate solution of the scalarized problem (6). It is in many ways similar to the first Algorithm 1, but uses a different decision vector $\boldsymbol{d}$. Note that indices p, q and r now count over hatch lines instead of over segments.

In the current implementation of the beam scanning, there are no pauses between any segments during melting. For instance, a jump from one hatch line to the next is instantaneous. However, adding delay time for jumps is straightforward.

6.1 Example 1: segmentwise optimization on snake pattern

We illustrate how the optimization scheme resolves certain heating related issues. The beam path is shown in Fig. 5. It consists of 50 segments, each with length 0.5 mm. The line offset during hatching is $l_{\text{off}} = 200\ \mu$m. The hatching is performed in a snake-like manner in the upward y-direction. The weight $\alpha(\boldsymbol{x})$ (see (5)) is zero on the first 0.4 mm and last 0.4 mm of a hatch line. While this beam path amounts to a simple rectangular shape, it still allows for several types of investigations.

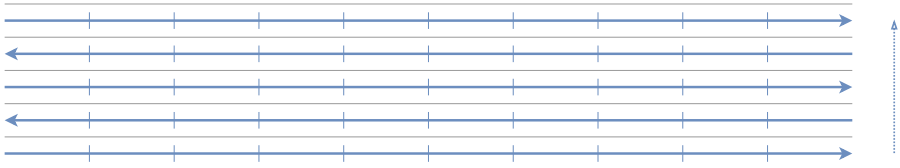


Figure 5 Beam path in Example 1: 5 hatch lines with length 5 mm. The path consists of 50 segments that are separated by the small ticks. The line offset (i.e., the distance between two adjacent lines) is 200 μm. The thin gray lines indicate the secondary path $\mathcal{C}^{\mathrm{wd}}$.

Table 1 Parameter values in Example 6.1.

PDE specific input			
Thermal conductivity	λ	20	(W/mK)
Thermal diffusivity	κ	8.45e–6	(m^2/s)
Initial temperature	u_{init}	1000	(K)
Absorbed beam power	P	100	(W)
Greedy algorithm specific input			
Reference surface temperature	u_{melt}	1800	(K)
Reference melt temperature	u_{surf}	2800	(K)
Secondary beam position, width	w	100	(μm)
Secondary beam position, depth	d	50	(μm)
Weight 1	W_1	0.7	
Weight 2	W_2	0.3	
Window size	$q-p+1$	5	
Segments frozen in each iteration	r	1	
Initial spot size on segment k	σ_k	0.2 $\forall k$	(mm)
Initial speed on segment k	v_k	0.5 $\forall k$	(m/s)
Bounds, spot size	$(\sigma_{\min}, \sigma_{\max})$	(1e–2, 1e0)	(mm)
Bounds, speed	$(v_{\min}, v_{\max})$	(1e1, 1e4)	(mm/s)

The secondary path is set to

$$\mathcal{C}^{\mathrm{wd}} = \left\{ \left(x^{\mathrm{s}}(t), y^{\mathrm{s}}(t) + w, -d \right) : t \in \mathcal{T} \right\}.$$

To avoid porosity, w and d need be chosen such that unmelted gaps between lines are avoided. Here we set $w = l_{\mathrm{off}}/2 = 100$ μm and $d = 50$ μm. A complete list of parameter values is given in Table 1. Figure 6 shows the solution of (6) as obtained by the greedy Algorithm 1. We see rapid variations in the optimized beam parameters, in particular close to the turning points where they seek to neutralize the concentrated influx of heat that occur in those regions. Problem (1) is then solved for the optimized beam parameters and the resulting maximum temperature in various slices of the domain is shown in Figs. 7, 8, and 9. These figures include the initial maximum temperature for comparison. The results indicate that despite the reductions leading up to its formulation, the greedy algorithm is able to control the heat generated during melting to a rather large degree.

One concern with the greedy algorithm is that it carries with it several uncertainties. Many trials are required to find proper values for the parameters that make up the scheme, such as α, the window size and segment lengths, since they depend on the beam path and thermal diffusivity.

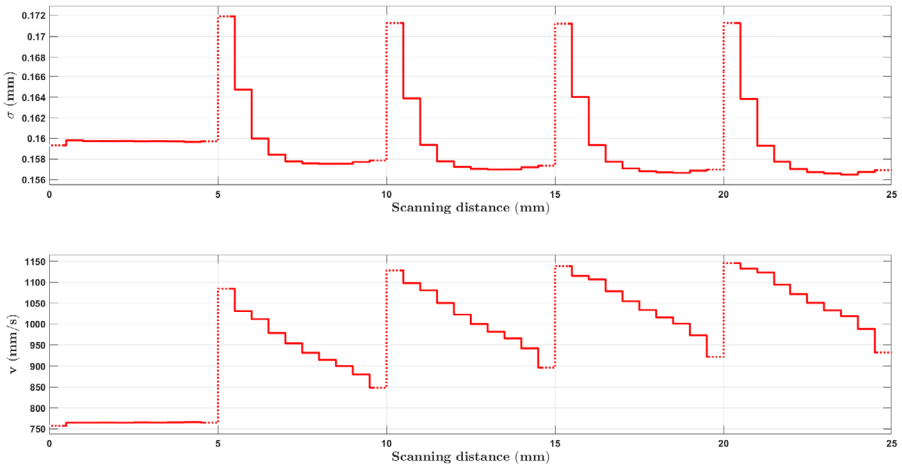


Figure 6 The solution as obtained by the first greedy algorithm. The dotted parts indicate the intervals where the weight $\boldsymbol{\alpha}$ is 0 (the start and end of the hatch lines).

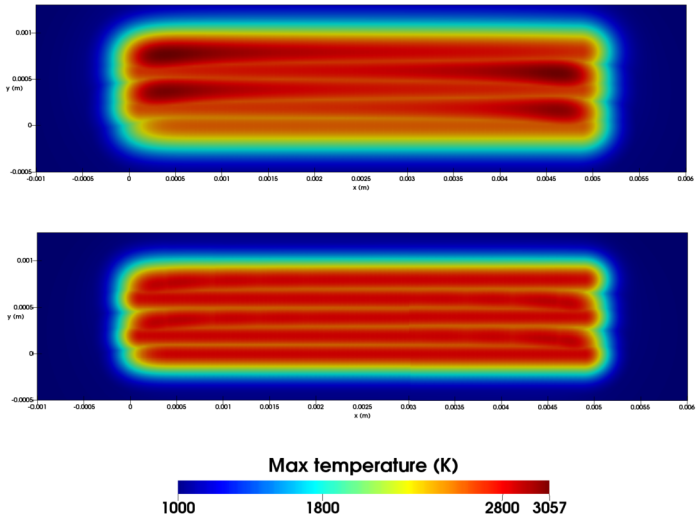


Figure 7 A comparison between maximum temperatures on the surface ($z = 0$) before optimization (top) and after optimization. The optimization scheme resolves the heat accumulation at the turning points.

6.2 Example 2: segmentwise optimization on nonparallel pattern

We melt the first quadrant of an annulus. The annulus has an inner radius $r_i = 1$ mm and outer radius $r_o = 5$ mm. The beam path is shown in Fig. 10. It consists of 19 lines of length 4 mm. Two adjacent lines differ by an angle of 5°. Consequently, the distance between them at r_i and r_o are about 87 μm and 436 μm, respectively. Each line is divided into 8 segments of equal length 0.5 mm. The hatching is performed in the counter-clockwise direction. The thin gray lines indicate the secondary path $\mathcal{C}^{\text{wd}}$. Once again, the weight $\alpha(\boldsymbol{x})$ is zero on the first 0.4 mm and last 0.4 mm of a hatch line. The remaining parameters are identical to the ones used in the previous example and are listed in Table 1.

Only the blue part of the beam path is considered during optimization. More precisely, we apply the greedy algorithm on the first 5 lines only. The results from this optimization

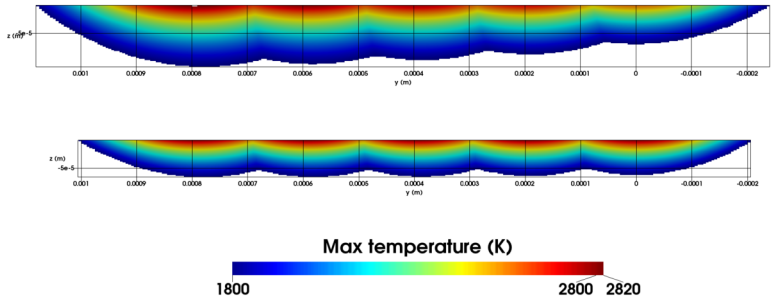


Figure 8 A comparison between maximum temperatures in the cross-section $x = 2.5$ mm before optimization (top) and after optimization. Initially, the melt area increases for each hatch line since the heat influx is larger than the rate of diffusion. The optimization scheme resolves this issue and makes the area more uniform.

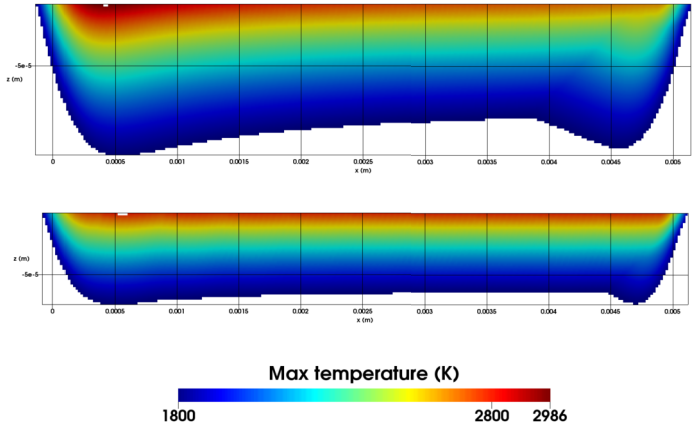
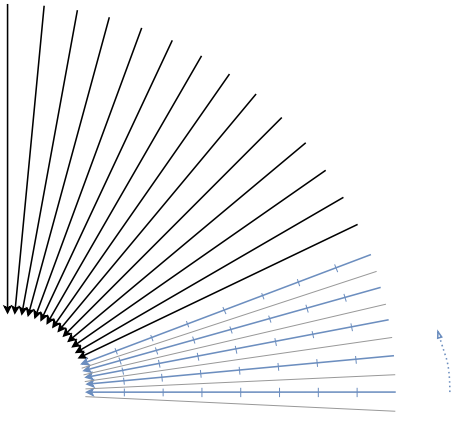


Figure 9 A comparison between maximum temperatures in the cross-section $y = 2\,l_{\text{off}} = 0.4$ mm (i.e., along the 3rd hatch line) before optimization (top) and after optimization. The depth of the melt area is reduced and made more uniform. The domain is scaled by a factor 10 in the vertical direction.

Figure 10 Beam path in Example 2 for melting the first quadrant of an annulus with $r_i = 1$ mm and outer radius $r_o = 5$ mm. The entire beam path consists of 19 lines of length 4 mm. Each line is divided into 8 segments of equal length 0.5 mm. Only the blue part, $\mathcal{C}^{\mathbf{s}}$, of the entire beam path is considered during optimization and the results from this optimization is then extended to the entire beam path. The path consists of 40 segments that are separated by the small ticks. The thin gray lines indicate the secondary path $\mathcal{C}^{\text{wd}}$.



is then extended by letting the beam parameters on lines 6-19 equal the optimal beam parameters on line 5.

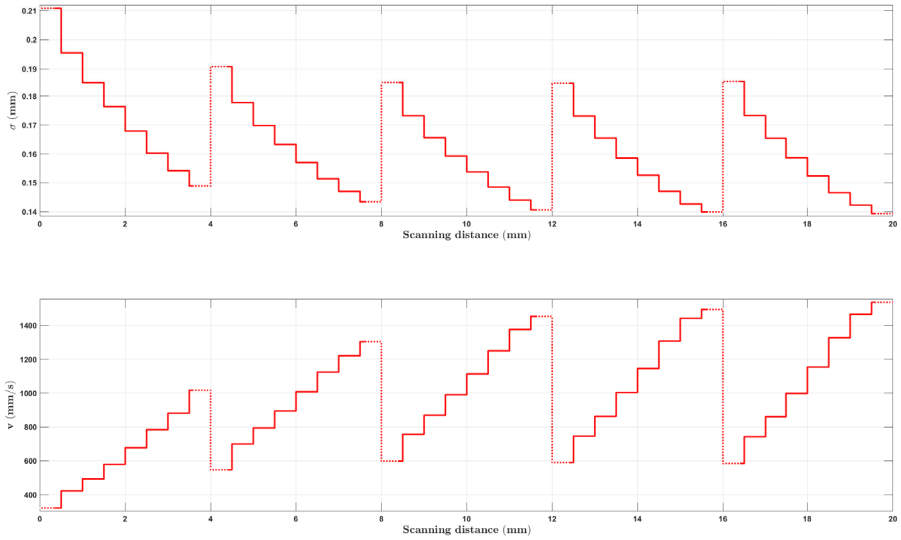


Figure 11 Optimal beam parameters in the second example, as obtained by the first greedy algorithm. The dotted parts indicate the places where the weight $\boldsymbol{\alpha}$ is 0 (the start and end of the hatch lines). The parameter values start to stabilize somewhat by the fifth line. This motivates us to copy the parameter values for the fifth line to the remaining lines 6-19 that were omitted in the optimization.

Figure 11 shows the solution of the optimization problem. The speed increases along each hatch line since the width between a hatch line and the corresponding secondary path decreases as the hatch line approaches $r = 1\,\text{mm}$ (see Fig. 10).

After having extended this solution to the entire beam path, problem (1) is solved for the optimized beam parameters and the resulting maximum temperature is shown in Figs. 12 and 13. These figures include the initial maximum temperatures for comparison.

The plots in Fig. 13 confirm that the extension of the solution onto remaining lines 6-19 works well. While the temperature on $\mathcal{C}^{\text{wd}}$ increases near the inner radius of the annulus as the scanning progresses, this increase is small and does not justify applying the greedy algorithm on the entire beam path, 19 lines, rather than just 5 lines. This is just a small example of how the results from the greedy algorithm on a very small section can be utilized on larger sections of the build area. In general, this procedure offers an efficient method for improving process control: first examine and optimize typical problematic melting scenarios, then combine the results and extend them to the remainder of the layer.

6.3 Example 3: linewise optimization on snake pattern

We now use the second greedy algorithm to solve the problem introduced in the first example, Sect. 6.1, and compare the results with the results obtained by the first greedy algorithm. Motivated by the solution obtained by the first greedy algorithm, see Fig. 6, we make the following ansatz. Let

$$\boldsymbol{\Lambda}_l = \left(\boldsymbol{\Lambda}_l^{\sigma}, \boldsymbol{\Lambda}_l^{v}\right) = \left(C_{1,l}^{\sigma}, C_{2,l}^{\sigma}, C_{3,l}^{\sigma}, C_{4,l}^{\sigma}, C_{1,l}^{v}, C_{2,l}^{v}, C_{3,l}^{v}, C_{4,l}^{v}\right)$$

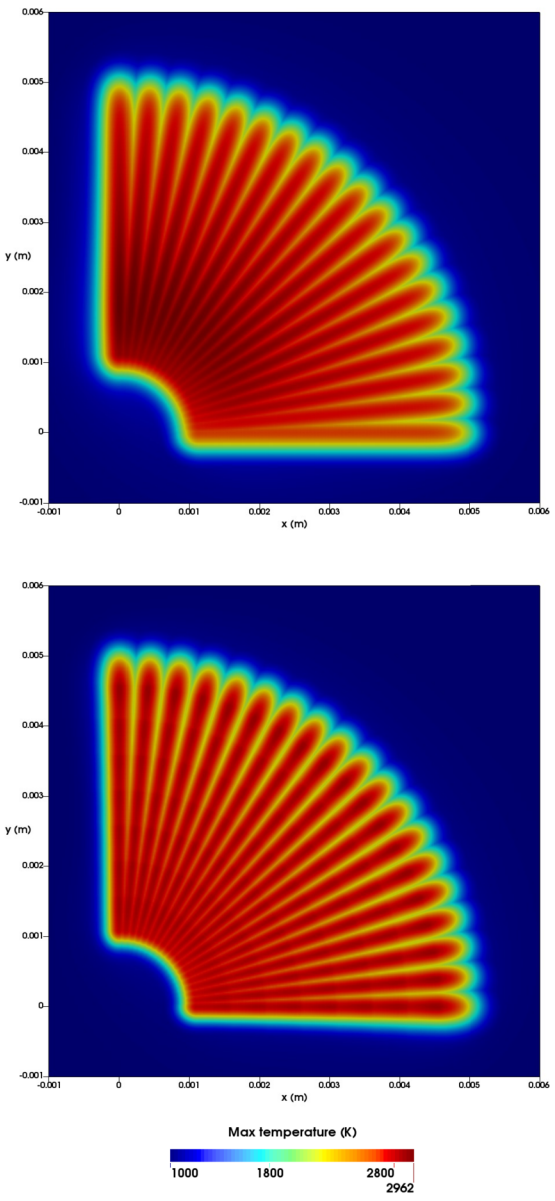


Figure 12 A comparison between maximum temperatures on the surface ($z = 0$) before optimization (top) and after optimization. The optimization scheme resolves the heat accumulation near the inner radius of the annulus. The optimized maximum surface temperature appears slightly jagged along the beam path, which suggests that the segment length of 0.5 mm is too big.

and

$$\begin{aligned} F_l^{\sigma}\left(\boldsymbol{x}^{\mathrm{s}};\boldsymbol{\Lambda}_l^{\sigma}\right) &= C_{1,l}^{\sigma}\left(1+\frac{C_{2,l}^{\sigma}}{1+C_{3,l}^{\sigma}(\gamma(\boldsymbol{x}^{\mathrm{s}})-\gamma(\boldsymbol{x}^{\mathrm{i}}_{\mathcal{S}(l)}))^{C_{4,l}^{\sigma}}}\right), \\ F_l^{\nu}\left(\boldsymbol{x}^{\mathrm{s}};\boldsymbol{\Lambda}_l^{\nu}\right) &= C_{1,l}^{\nu}\left(1+\frac{C_{2,l}^{\nu}}{1+C_{3,l}^{\nu}(\gamma(\boldsymbol{x}^{\mathrm{s}})-\gamma(\boldsymbol{x}^{\mathrm{i}}_{\mathcal{S}(l)}))^{C_{4,l}^{\nu}}}\right), \end{aligned} \tag{9}$$

for $l = 1, \ldots, M$. Hence we associate 8 coefficients with each hatch line. From the beam path in Fig. 5, we have $M = 5$. We solve optimization problem (6) according to Algorithm 2. The

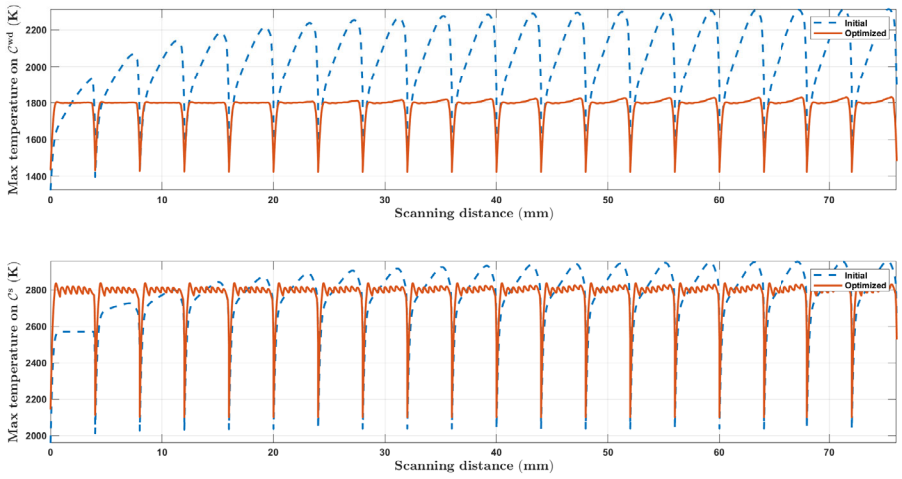


Figure 13 A comparison between maximum temperatures on the (extended) secondary path and (extended) beam path. The reference temperatures $u_{\text{melt}} = 1800$ K and $u_{\text{surf}} = 2800$ K are successfully tracked.

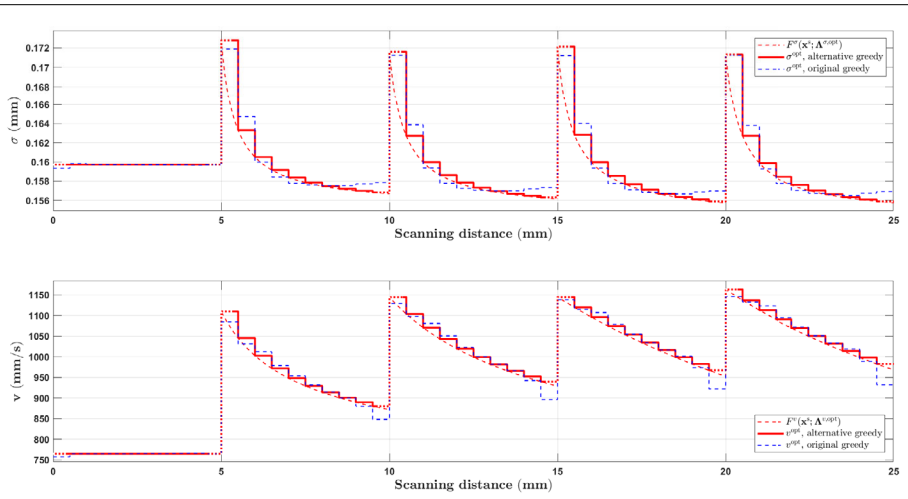


Figure 14 The solution as obtained by the second greedy algorithm. The solution is a piecewise constant discretization of the beam parameter functions F^{σ} and F^{v} in (9). The solution obtained by the first greedy algorithm is included for comparison.

window always consist of 1 hatch line, i.e., 10 segments. The results are shown in Fig. 14. The solution obtained by the first greedy algorithm is included for comparison. The results are similar. The corresponding optimal objectives J^{opt} are similar as well, as $J^{\text{opt}} = 124.18$ with the second greedy algorithm and $J^{\text{opt}} = 123.03$ with the first greedy algorithm ($J^{\text{init}} = 1655.21$).

We end this section with a comparison between the first greedy algorithm and the second greedy algorithm. The first algorithm from Sect. 4 optimizes the beam parameters segmentwise. It can be applied to general beam paths and there are no restrictions on the window. In particular it and can be used to look at specific problematic areas of the layer being melted in order to get an understanding of how the beam parameter functions should behave in those areas.

The second greedy algorithm from Sect. 5 optimizes the beam parameters linewise. This makes the second algorithm preferable in practical problems because it significantly decreases the dimension of the decision space; the number of lines M is much lower than the number of segments N. In the above example, the hatch lines are fairly short, but we still get $40 = 8M < 2N = 100$ when comparing dimensions of the two decision spaces. For more realistic problems the number of segments could be orders of magnitude larger, making the second greedy algorithm more attractive.

The potential drawback of the second algorithm is that it might be difficult to make the initial ansatz for the beam parameter functions since they depend on the beam path (and secondary path). However, this is where the first algorithm can be utilized to give an initial estimate that shows the behavior of the desired beam parameters. This approach is what enabled us to choose the beam parameter functions in (9). For the future, we imagine the development of a database of parameter functions that have been generated for different melting scenarios and that can be shared and used in other, more detailed models.

7 Conclusions

We have formulated an optimization scheme for controlling the heat conduction during the melting process in powder-bed-based additive manufacturing. The scheme is efficient because it exploits that the melt pool is local to the beam and shows good capabilities despite several simplifications. The current choice of objectives prioritizes speed since it only requires temperature evaluations on lines rather than in entire volumes. The scheme should be useful for studying problematic areas of the melting process where particular care needs to be put into the choice of beam parameters.

The optimization scheme relies on a greedy algorithm. Two versions of a greedy algorithm have been presented and applied in this paper. The first one carries out optimization segmentwise, which makes it applicable to many types of beam paths. The second one carries out optimization linewise, which can significantly reduce the dimension of the decision space. While the two algorithms are similar, they serve different purposes and we have detailed how they can be combined to improve process control.

The examples considered in this paper are fairly small. For more realistic problems the number of segments could be orders of magnitude larger. By design, the greedy algorithm becomes more attractive as the total amount of segments in the beam path increases; the division of the global problem (6) into subproblems (7) becomes more beneficial, relatively speaking, as N (and M) increases. Furthermore, the examples are purely numerical. It is currently difficult to compare the results in Sect. 6 with experiments because existing machines lack the functionality required to match our numerical results. Because of this, a crucial next step is to implement the necessary code in the machine such that experimental validation becomes possible. Experiments are also needed for the generation of effective parameters; the analytic model is very simple and since it contains few parameters, they need to be carefully fit against experiments.

The optimization method also needs to be complemented with different types of testing. It requires effective reference temperatures u_{melt}, u_{surf}. Furthermore, the weights and the secondary path $\mathcal{C}^{\text{wd}}$ need to be carefully chosen. Work related to this kind of testing has not been detailed here.

Acknowledgements
The authors are grateful to the anonymous referee for constructive criticism.

Funding
This work was supported by the Swedish Foundation for Strategic Research under the contract "Industrial PhD 2015 – ID15-0058" and by the H2020 European funded project EBMPerform no. 666788. Open access funding provided by Chalmers University of Technology

Abbreviations
AM, Additive Manufacturing; DoE, Design of Experiments; PBF, Powder Bed Fusion.

Availability of data and materials
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Authors' contributions
The main idea of this paper was proposed by AS. RF wrote the manuscript, implemented the algorithm and ran the numerical examples. RF, AS and SL closely reviewed and discussed all stages of development. All the authors read and approved the final manuscript.

Publisher's Note

Received: 7 February 2019 Accepted: 4 August 2021 Published online: 19 August 2021

References

1. Heinl P, Müller L, Körner C, Singer RF, Müller FA. Cellular Ti–6Al–4V structures with interconnected macro porosity for bone implants fabricated by selective electron beam melting. Acta Biomater. 2008;4(5):1536–44. https://doi.org/10.1016/j.actbio.2008.03.013.
2. Mani M, Lane B, Donmez MA, Feng S, Moylan S, Fesperman R. Measurement science needs for real-time control of additive manufacturing powder bed fusion processes. Technical report. Gaithersburg, MD: National Institute of Standards and Technology; 2015. NIST Interagency/Internal Report (NISTIR).
3. King WE, Anderson AT, Ferencz RM, Hodge NE, Kamath C, Khairallah SA, Rubenchik AM. Laser powder bed fusion additive manufacturing of metals; physics, computational, and material challenges. Appl Phys Rev. 2015;**2**(4). https://doi.org/10.1063/1.4937809.
4. Markl M, Körner C. Multi-scale modeling of powder-bed-based additive manufacturing. Annu Rev Mater Res. 2016;46:93–123. https://doi.org/10.1146/annurev-matsci-070115-032158.
5. Zeng K, Pal D, Stucker B. A review of thermal analysis methods in laser sintering and selective laser melting. In: Solid freeform fabrication symposium. 2012. p. 796–814.
6. Ma L, Fong J, Lane B, Moylan S, Filliben J, Heckert A, Levine L. Using design of experiments in finite element modeling to identify critical variables for laser powder bed fusion. In: Solid freeform fabrication symposium. 2015. p. 219–28.
7. Kamath C, El-dasher B, Gallegos GF, King WE, Sisto A. Density of additively-manufactured, 316L SS parts using laser powder-bed fusion at powers up to 400 W. Int J Adv Manuf Technol. 2014;74(1):65–78. https://doi.org/10.1007/s00170-014-5954-9.
8. Eagar TW, Tsai N-S. Temperature fields produced by traveling distributed heat sources. Weld Res Suppl. 1983;62:346–55.
9. Ning Y, Fuh JYH, Wong YS, Loh HT. An intelligent parameter selection system for the direct metal laser sintering process. Int J Prod Res. 2004;42(1):183–99. https://doi.org/10.1080/00207540310001595873.
10. Garg A, Tai K, Savalani MM. State-of-the-art in empirical modelling of rapid prototyping processes. Rapid Prototyping J. 2014;20(2):164–78. https://doi.org/10.1108/RPJ-08-2012-0072.
11. Hinze M, Ziegenbalg S. Optimal control of the free boundary in a two-phase Stefan problem. J Comput Phys. 2007;223(2):657–84. https://doi.org/10.1016/j.jcp.2006.09.030.
12. Volkov O, Protas B, Liao W, Glander DW. Adjoint-based optimization of thermo-fluid phenomena in welding processes. J Eng Math. 2009;65(3):201–20. https://doi.org/10.1007/s10665-009-9292-0.
13. Cao X, Ayalew B. Partial differential equation-based multivariable control input optimization for laser-aided powder deposition processes. ASME J Manuf Sci Eng. 2015;138(3):031001. https://doi.org/10.1115/1.4031265.
14. Malmberg JB, Wallenås M. Solving the heat equation in connection with electron beam melting. Master's thesis. Department of Mathematical Sciences, Mathematics, Chalmers University of Technology; 2012. http://publications.lib.chalmers.se/records/fulltext/159984.pdf.
15. Forslund R, Snis A, Larsson S. Analytical solution for heat conduction due to a moving Gaussian heat flux with piecewise constant parameters. Appl Math Model. 2019;**66**. https://doi.org/10.1016/j.apm.2018.09.018.
16. Snis A. Method for production of a three-dimensional body. 2015. US Patent 9,073,265 B2.
17. Khairallah SA, Anderson AT, Rubenchik A, King WE. Laser powder-bed fusion additive manufacturing: physics of complex melt flow and formation mechanisms of pores, spatter, and denudation zones. Acta Mater. 2016;108:36–45. https://doi.org/10.1016/j.actamat.2016.02.014.
18. King WE, Anderson AT, Ferencz RM, Hodge NE, Kamath C, Khairallah SA. Overview of modelling and simulation of metal powder bed fusion process at Lawrence Livermore National Laboratory. Mater Sci Technol Ser. 2015;31(8):957–68. https://doi.org/10.1179/1743284714Y.0000000728.
19. Mukherjee T, Zuback JS, De A, DebRoy T. Printability of alloys for additive manufacturing. Sci Rep. 2016;**6**. https://doi.org/10.1038/srep19717.
20. Gong H, Rafi K, Starr T, Stucker B. The effects of processing parameters on defect regularity in Ti-6Al-4V parts fabricated by selective laser melting and electron beam melting. In: Solid freeform fabrication symposium. 2013. p. 424–39.

21. Vandenbroucke B, Kruth J. Selective laser melting of biocompatible metals for rapid manufacturing of medical parts. Rapid Prototyping J. 2007;13(4):196–203. https://doi.org/10.1108/13552540710776142.
22. Smith CJ, Derguti F, Nava EH, Thomas M, Tammas-Williams S, Gulizia S, Fraser D, Todd I. Dimensional accuracy of Electron Beam Melting (EBM) additive manufacture with regard to weight optimized truss structures. J Mater Process Technol. 2016;229:128–38. https://doi.org/10.1016/j.jmatprotec.2015.08.028.
23. Zhu C, Byrd RH, Lu P, Nocedal J. Algorithm 778: L-BFGS-b: fortran subroutines for large-scale bound-constrained optimization. ACM Trans Math Softw. 1997;23(4):550–60. https://doi.org/10.1145/279232.279236.
24. Byrd RH, Lu P, Nocedal J, Zhu C. A limited memory algorithm for bound constrained optimization. SIAM J Sci Comput. 1995;16(5):1190–208. https://doi.org/10.1137/0916069.
25. Jones E, Oliphant T, Peterson P et al. SciPy: open source scientific tools for python. 2001. http://www.scipy.org/.
26. Lewis AS, Overton ML. Nonsmooth optimization via quasi-Newton methods. Math Program. 2013;141(1):135–63. https://doi.org/10.1007/s10107-012-0514-2.